

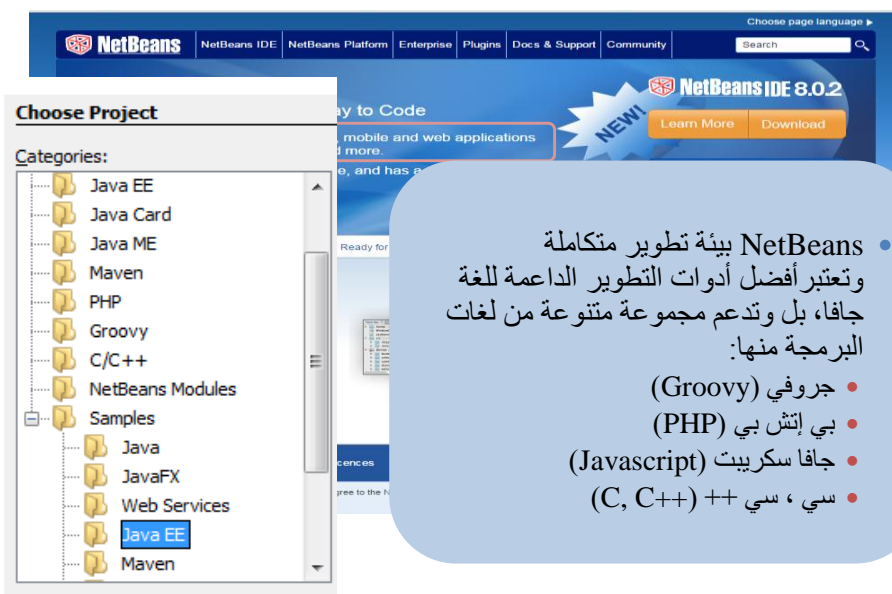
## البرمجة الشيئية

# Object Oriented Programming (with Java) ITGS211

## المحاضرة الثانية

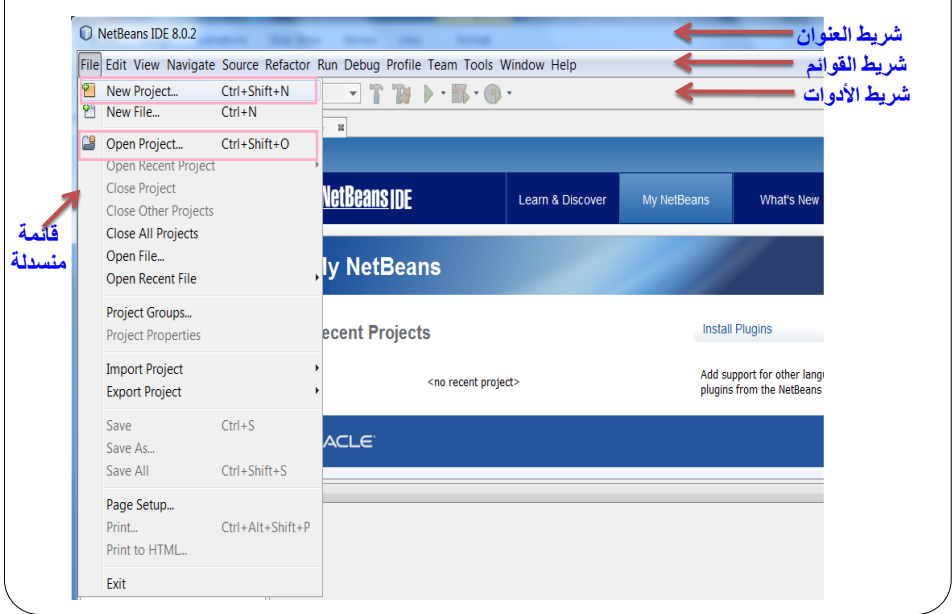
الفصل الدراسي: ربيع 2017

## التعرف على أداة التطوير NetBeans

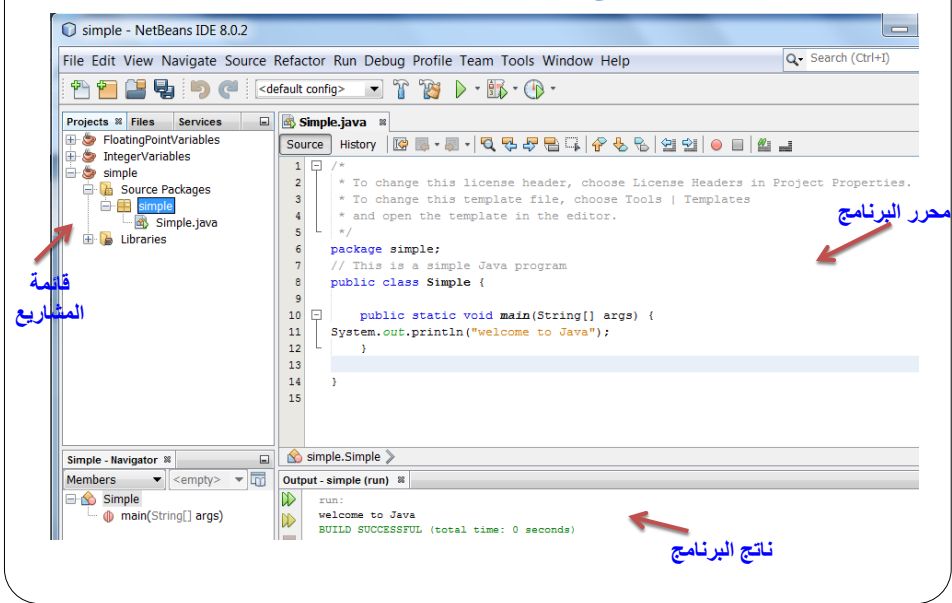


- NetBeans بيئة تطوير متكاملة وتعتبر أفضل أدوات التطوير الداعمة للغة جافا، بل وتدعم مجموعة متنوعة من لغات البرمجة منها:
  - جروفي (Groovy)
  - بي إتش بي (PHP)
  - جافا سكريبت (Javascript)
  - سي ، سي ++ (C, C++)

# التعرف على أداة التطوير NetBeans

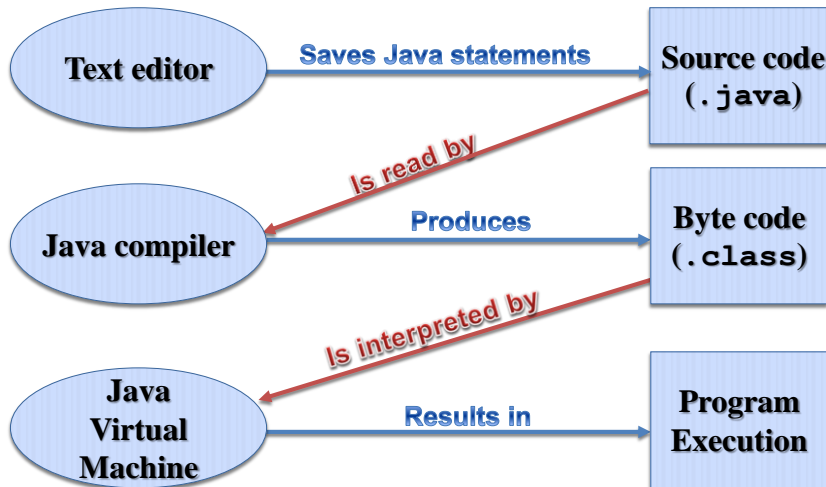


# التعرف على أداة التطوير NetBeans

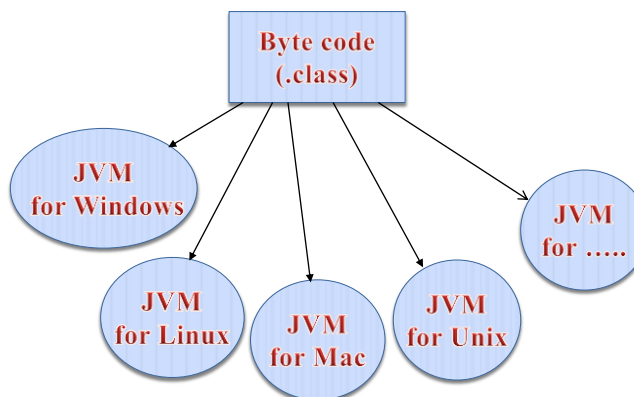


## برنامج جاوا

الجهاز لا يفهم أوامر هذه اللغة، لذا يجب ترجمتها إلى لغة الآلة حسب الخطوات التالية:



تتمتع برامج Java بقابلية النقل Portability والتي تعني أن البرنامج يُكتب على نوع معين من أجهزة الحاسوب، مع إمكانية نقله وتشغيله على مجموعة من أنواع أخرى، مع تعديل ضئيل (و قد يكون معدوم). أي أنها لا تعتمد على نظام تشغيل محدد Platform Independent؛ والسبب هو أن الـ Byte code يعمل على JVM وليس على CPU الخاصة بالحاسوب مباشرة. والـ JVM متضمنة في العديد من الـ platforms.



## مكونات برنامج جافا

```
// This is a simple Java program.
```

This is a Java comment.  
It is ignored by the compiler.

```
public class Simple
```

This is the class header  
for the class Simple

```
{
```

This area is the body of the class Simple.  
All of the data and methods for this class  
will be between these curly braces.

```
}
```

## مكونات برنامج جافا

```
// This is a simple Java program.
```

This is the method  
header for the main  
method.  
The main method is  
where a Java application  
begins.

```
public class Simple
{
```

```
    public static void main(String[] args)
```

```
    {
```

This area is the body of the main method.  
All of the actions to be completed during  
the main method will be between these curly braces.

```
    }
```

```
}
```

### مكونات برنامج جافا

**Key words** كلمات محجوزة

**Class header**

**Method header**

**Comment** تعليق

```

public class HelloWorld {
    // this is a HelloWorld program
    public static void main(String[] args) {
        // TODO code application logic here
        String message = "Hello World";
        System.out.println(message);
    }
}

```

**Output** ناتج البرنامج

run: Hello World

### مكونات برنامج جافا

✓ **التعليقات Comments:**

- أسطر التعليق يتم تجاهلها من قبل الـ compiler .
- المثال السابق يحوي تعليقين كل منهما عبارة عن سطر واحد - single line لذلك تم استخدام `//`.

✓ **الـ class:** واحد أو أكثر، منها واحد فقط يكون **Public**.

Class Header يُعلم الـ Compiler ببعض الأمور عن الـ Class مثلاً:

```

public class HelloWorld {
}

```

- **Public** أمكانية استخدام الـ **Classes** الأخرى لهذا الـ Class .
- **class** أنه يعتبر Java class.
- HelloWorld هو اسم الـ class، وعند حفظ ملف جافا لابد ان يتم حفظه بنفس اسم الـ class وب نفس شكل الحروف HelloWorld.java.


## مكونات برنامج جافا

```
Public Static void main (Strings [] args ){
    :
}
```

- **void** تعني بأن الدالة بعد تنفيذ البرنامج لن تعود بأي قيم.
- **main** تعتبر نقطة البداية لوظيفة، كما في لغة C++
- **Strings [ ] args** الجملة الموجودة داخل قوس البداية للدالة **main** تعني مصفوفة من النوع الحرفي لتخزين جملة الطباعة في البرنامج.
- { قوس بداية الدالة.
- } يتم انتهاء الدالة **main** بقوس النهاية.
- ✓ **أقواس المجموعة { }** **Curly Braces**
- عندما يقترن بـ (class header)، فإنه يحدد مدى الرؤية (Scope) للـ (Class)
- عندما يقترن بـ (method)، فإنه يحدد مدى رؤية الـ (method).

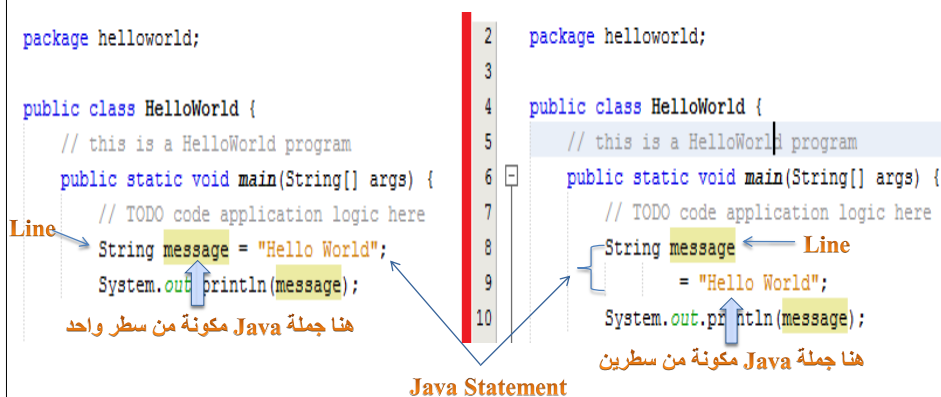
## ملاحظات حول برنامج جافا

- لغة Java تعتبر حساسة لحالة الأحرف (case-sensitive).
- المترجم يتجاهل كل التعليقات Comments.
- يجب أن يتم تخزين جميع برامج جافا في ملف بملحق (.java).
- اسم ملف (.java) هو نفس اسم الـ **public class**، في البرنامج. كما في المثال السابق.

`public class HelloWorld` →  HelloWorld.java

- تطبيقات جافا Java applications لابد أن تحوي **main** method
- كل قوس أيسر { left brace لابد أن يقابله قوس أيمن right brace }

► كل جملة Statement (وليس سطر) في جافا يجب أن تنتهي بفاصلة منقوطة (;).



**Line ≠ Statement**

► التعليقات Comments و class headers و method headers و java Statement لا تعتبر لذلك لا تنتهي بفاصلة منقوطة (;) والأقواس

### مفردات لغة جافا

- لكل لغة رموزها ومصطلحاتها الخاصة بها. والتي قد تختلف من لغة لأخرى.

- الأبجدية الإنجليزية Letters :

- الحروف الصغيرة a,b, .....,y,z
- الحروف الكبيرة A,B,.....,Y,Z

- الأرقام Numbers :

- من 0 إلى 9

- الرموز الخاصة Special Symbols :

" + - \* / , ; . ( ) \$ % ^ ?

## الكلمات المحجوزة Reserved Words

- الكلمات المحجوزة هي التي لا يمكن استخدامها كاسم متغير (Variable) أو فصيل (Class) أو طريقة (Method) في البرنامج لان لها معنى خاص للمترجم (Compiler).
- الكلمات المحجوزة في لغة Java هي:

abstract	else	interface	switch
assert	enum	long	synchronized
boolean	extends	native	this
break	false	new	throw
byte	final	null	throws
case	finally	package	transient
catch	float	private	true
char	for	protected	try
class	goto	public	void
const	if	return	volatile
continue	implements	short	while
default	import	static	
do	instanceof	strictfp	
double	int	super	

## التعليقات Comments

- التعليق هو رسالة توضيحية لتوثيق وشرح البرنامج والهدف منه، ويمكن كتابتها في أي مكان من البرنامج شرط أن:
- تبدأ ب // وتنتهي بنهاية السطر لو كانت من سطر واحد، كما يلي.
- // This is My First program in Java
- تبدأ ب /\* وتنتهي ب \*/ لو كانت أكثر من سطر. كما يلي.

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javaapplication1;

public class JavaApplication1 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int a= 20;
    }
}

```

- المترجم (Compiler) يتجاهل كل التعليقات.



## الثوابت

- الثابت: هو موقع تخزيني مؤقت في الذاكرة يستخدم لتخزين قيمة ثابتة لا تتغير أثناء تنفيذ البرنامج . وتنقسم الثوابت إلى :

### ➤ ثوابت عددية Numeric Constants

A . الثابت العددي الصحيح Integer constant

B . الثابت العددي الحقيقي Floating constant

### ➤ ثوابت رمزية Non-Numirec Constants

## الثابت العددي الصحيح Integer constant

- عبارة عن عدد مكون من الأرقام من (0-9) .
- لا يحتوي على فاصلة عشرية.
- يمكن أن يحتوي على إشارة ( + أو - ) .
- كما يمكن تصنيف الأعداد الصحيحة حسب طولها والسعة التخزينية لها في الذاكرة ومن الأمثلة

( 12، 100، - 30، ..... ) .

## الثابت العددي الحقيقي Floating constant

- عبارة عن عدد مكون من الأرقام من (0-9) .
- يجب أن يحتوي على فاصلة عشرية.
- يمكن أن يحتوي على إشارة ( + أو - ) .
- من الأمثلة على الأعداد الحقيقية :

( 13.2، -438.05، 3.50، ..... ) .

## ثوابت الرمزية

❖ عبارة عن رموز اللغة وتتكون من الحروف والأرقام وتكون بين علامتي تنصيص أو أقتباس ومن الأمثلة : ( " 1324 " ، "Khal+ed" ، "name" ، ..... )  
ونلاحظ أن هناك ثوابت رمزية تحتوي على أرقام وهي في الحقيقة قيم حسابية لا يمكن إجراء أي عملية حسابية عليها بل يتم طبع الأرقام أو الرموز وإذا أردنا أن نطبع قيم سوف تظل ثابتة داخل البرنامج في مكان في الذاكرة فإننا نستخدم العبارة Final للإعلان أن هذه القيمة ستظل ثابتة طول تنفيذ البرنامج مثل :

```
Final int TABLE_SIZE= 41;
```

```
Final float PI =3.14;
```

## المتغيرات Variables

- هو اسم لموقع تخزيني مؤقت في الذاكرة يستخدم لتخزين قيمة داخلية ويختلف نوع المتغير حسب نوع القيمة المراد تخزينها وهناك مجموعة كبيرة من أنواع المتغيرات .

### قواعد تسمية المتغيرات

- 1- يمكن أن يبدأ الاسم بالحرف أو الشرطة السفلية (under score) أو علامة الدولار (\$) ولكن لا يمكن بدء التسمية برقم ولكن يمكن أن نضع رقم بعد الحرف
- 2- لا يمكن تسمية المتغير بإحدى الكلمات المحجوزة .
- 3- اسم المتغير يكون بالأحرف الصغيرة لكل الأحرف ويلاحظ عدم وجود أقواس .

• تنقسم المتغيرات إلى نوعين :

### 1- متغيرات رمزية ( حرفية )

تتضمن الحروف بكافة أشكالها والرموز والفراغات (مسافة فارغة)، وتتكون من خانة واحدة فقط محصورة بين علامتي تنصيص فردية (‘ ‘) مثل :

char a; a='F'

~~Char a,b; a="aaah"; b="4225drft"~~

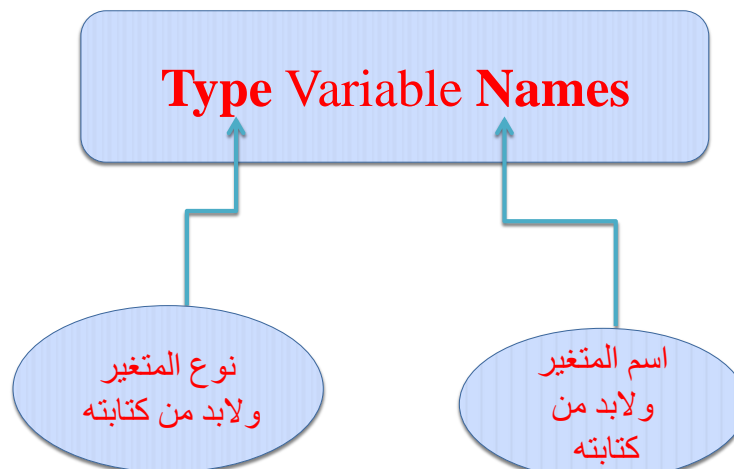
### 2- متغيرات عددية

تتضمن قيم عددية صحيحة يمكن أن نأخذ قيمة تصل إلى (32767) وتكتب على الشكل التالي :

inta; a=100 ;

int value ; value=2\*3

## كيفية تعريف المتغير



## Variable.java

مثال:

```
// This program has a variable.

public class Variable
{
    public static void main(String[] args)
    {
        int value;

        value = 5;
        System.out.print("The value is ");
        System.out.println(value);
    }
}
```

## Variables and Literals

**int value;**  
الاعلان عن المتغير **value**  
(variable declaration)

**value = 5;**  
جملة تعيين أو اسناد  
assignment statement.



السلسلة الحرفية (string literal) " " ستطبع كما هي نظراً لوجود " " هنا ستطبع القيمة الصحيحة 5 لاحظ عدم وجود " "

```
System.out.print("The value is ");
System.out.println(value);
```

## بعض تسمية المتغيرات والاعلان عنها

int name1 , name2 ; // ok

long good-by ; // خطأ لأنه يحتوي على الشرطة

short shrift =0 ; // ok

byte the mark ; // خطأ لأنه يحتوي على مسافة

int float ; // خطأ لوجود كلمة محجوزة

char thismustbetoolong ; // خطأ اسم متغير طويل جداً

int 6vsdgtf ; // خطأ لأنه بدأ برقم

float a=12.3 ; b=5.78 ; c=44.9 ; // خطأ لاستخدام الفاصلة المنقوطة لفصل المتغيرات

## بعض تسمية المتغيرات والاعلان عنها

- لاحظ أن : رغم أنواع المتغيرات Types تكتب بالحروف الصغيرة ، إلا أن النوع String يبدأ بحرف كبير. فهو Class وليس نوع من أنواع البيانات الأساسية في اللغة.
- فلو بدأت بحرف صغير ستظهر رسالة تنبيه لخطأ لغوي

```
public class HelloWorld {
    // this cannot find symbol
    public symbol: class string
    // location: class HelloWorld
    int ----
    fin (Alt-Enter shows hints)
}

string message = "Hello World";
System.out.println(message + " ,pi= " + pi);
```

## أنواع البيانات و المتغيرات

المتغير	الحجم	نوع المتغير
byte	1 byte = 8 bits	متغيرات صحيحة
short	2 bytes = 16 bits	
int	4 bytes = 32 bits	
long	8 bytes = 64 bits	
float	4 bytes = 32 bits	متغيرات كسرية
double	8 bytes = 64 bits	
char	2 bytes = 16 bits	متغيرات نصية
String	-	
Boolean	1 bit	المتغير المنطقي

## العمليات الحسابية

EX	الوصف	المعامل
$X=A+B$	جمع	+
$X=A-B$	طرح	-
$X=A*B$	ضرب	*
$X=A/B$	قسمة	/
$(A+=B) = (A=A+B)$	جمع ثم اسناد	+=
$(A-=B) = (A=A-B)$	طرح ثم اسناد	-=
$(A*=B) = (A=A*B)$	ضرب ثم اسناد	*=
$(A/=B) = (A=A/B)$	قسمة ثم اسناد	/=
$(A\%=B) = (A=A\%B)$	باقي القسمة ثم اسناد	\%=
$A++ = A=A+1$	زيادة بمقدار واحد	++
$A-- = A=A-1$	نقصان بمقدار واحد	--
$X=A\% B$	باقي القسمة	%

العمليات المنطقية

EX	شكل العملية	المعامل بالشكل الرياضي
X==Y	=	=
X !=Y	!=	≠
X >Y	>	>
X <Y	<	<
X <=Y	<=	≤
X >=Y	>=	≥
If (X==1 && y==1)	& او &&	And
if(X==1    Y==1)	أو	Or
If (X==1 ^ y==1)	^	XOR

رموز الهروب Escape Characters

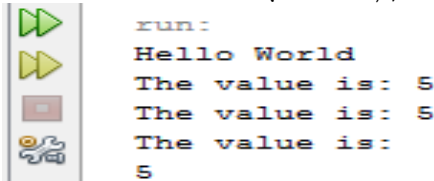
\n	newline	القفز إلى السطر التالي لتبدأ عنده الطباعة القادمة
\t	tab	المؤشر يتقدم tab (أي سبع مسافات للأمام) قبل الطباعة
\b	backspace	المؤشر يرجع مسافة واحدة للخلف (أي إلى اليسار)
\r	carriage return	المؤشر يرجع لبداية السطر الحالي وليس السطر التالي.
\\	backslash	يطبع الشرطة المائلة \
\'	single quotation	يطبع علامة تنصيص فردية ' single quotation mark
\"	double quotation	يطبع علامة تنصيص مزدوجة " double quotation mark

## المؤثر +

- نركز على المؤثر + حيث أن له خصوصية فبالإمكان استخدامه بطريقتين:

- (1) رابط مع السلاسل concatenation operator .
- (2) معامل جمع مع الأعداد an addition operator .
- فإذا كان ما على طرفي المعامل (+) سلسلتين فهو سيعمل على ربط السلسلتين وينتج سلسلة string.

```
int value=5;
System.out.println("Hello " + "World");
System.out.println("The value is: " + 5);
System.out.println("The value is: " + value);
System.out.println("The value is: " + '\n' + 5);
```



```
run:
Hello World
The value is: 5
The value is: 5
The value is:
5
```

## أولويات تنفيذ المؤثرات Operator Precedence

- قد يكون التعبير الرياضي معقداً جداً، لذلك وجب وجود أولويات لتنفيذ العمليات الحسابية

	Operator	Associativity	Example	Result
Higher Priority	- (unary negation)	right to left	x = -4 + 3;	-1
	* / %	left to right	x = -4 + 4 % 3 * 13 + 2;	11
Lower Priority	+ -	left to right	x = 6 + 3 - 4 + 6 * 3;	23

- عندما يتم استخدام الأقواس في تعبير، تتم معالجة الأقواس الداخلية أولاً.
- إذا وجد مجموعتين من الأقواس في نفس المستوى، تتم المعالجة من اليسار إلى اليمين.
- إذا وجدت عمليات من نفس المستوى تتم المعالجة من اليسار إلى اليمين.



```
int x,c;
```

• **x = ((4\*5) / (5-2)) - 25;**  $\Rightarrow$  result=-19  
 $= (20 / 3) - 25$   
 $= (6) - 25$   
 $= -19$

• **C = 3\*7-6+2\*5/4+6;**  $\Rightarrow \Rightarrow$  result=23  
 $= ((3*7)-6) + ((2*5)/4) + 6$   
 $= ((21-6) + (10/4)) + 6$   
 $= ((21-6) + 2) + 6$   
 $= (15 + 2) + 6$   
 $= 17 + 6$   
 $= 23$

**انتهت المحاضرة  
أي اسئلة أو استفسارات**

